DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment: 1

Student Name: Divyansh
UID: 23BCS11778
Branch: BE-CSE
Section/Group: KRG-2A
Semester: 6th
Date of Performance: 14/01/2026
Subject Name: System Design
Subject Code: 23CSH-314

Aim

To design an E-commerce platform similar to Amazon or Flipkart that enables users to search, view, purchase, and track products by identifying functional and non-functional requirements and defining system APIs.

Objectives

- Understand the architecture of a large-scale E-commerce system.
- Identify functional requirements such as product search, cart operations, and checkout.
- Identify non-functional requirements like scalability, consistency, and availability.
- Analyze race conditions during flash sales.
- Design RESTful APIs for core E-commerce operations.

Procedure

1. Studied real-world E-commerce platforms like Amazon and Flipkart.
2. Identified core entities such as User, Product, Cart, Order, and Payment.
3. Analyzed user flow from product search to order placement.
4. Collected functional and non-functional requirements.
5. Designed APIs for search, cart, checkout, payment, and order status.
6. Applied pagination to reduce frontend latency.
7. Studied race conditions during limited inventory scenarios.

Functional Requirements

1. Search products by title or name.
2. View product details including description, image, quantity, and reviews.
3. Add, update, and remove products from cart.
4. Perform checkout and payment.
5. Track order status.
6. Handle limited inventory purchases.
7. Prevent race conditions during flash sales.

Core Entities of the System

- User / Client
- Product
- Cart
- Orders
- Checkout and Payment

API Design

1. Product Search API

GET

https://localhost/products/search item={search keywords} GET:

<iPhone 16>

List<ProductID: iPhone>

Pagination is used to reduce latency.

2. View Product Details API

GET

https://localhost/products/{product id}

Product_id: 17
Name: iPhone 17
Color: Navy Blue
Price: $1099
Image Thumbnail: URL_Image

3. Add Item to Cart API

POST

https://localhost/cart/add products

Product_id = 17
Product_id = 16
User_id: 04

Cart_id: 101

4. Update Cart API – PUT

5. Remove Item from Cart API – DELETE

6. Checkout and Payment API – POST

https://localhost/checkout https://
localhost/payment

All Product IDs
Total Quantity
Total Price

Order_ID

Payment Type
Payment Mode

Confirmation_Status: Success / Fail

7. Order Status API – GET
https://localhost/order status={order id}

Non-Functional Requirements

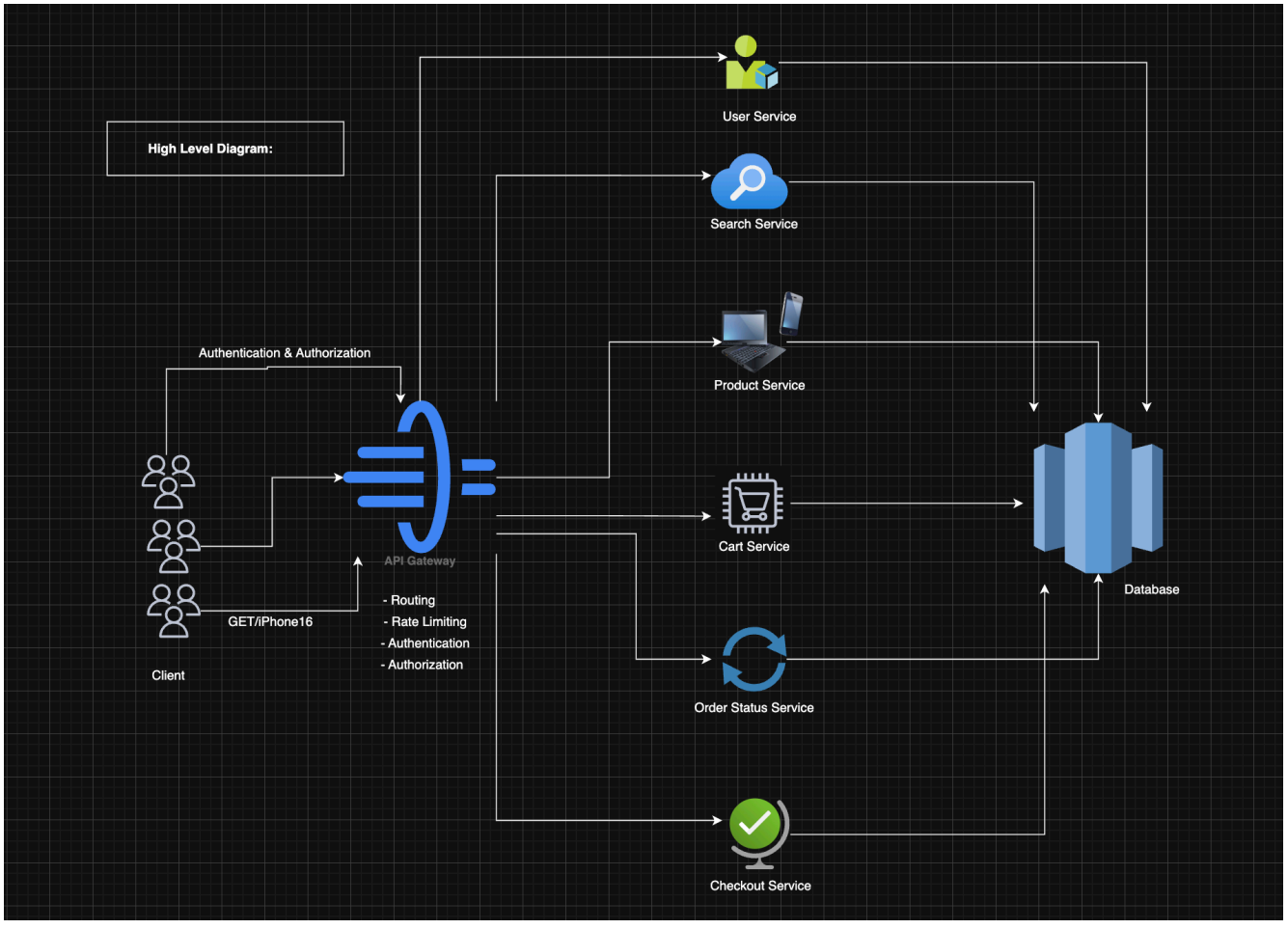
1. 100 million daily active users.
2. 10 orders processed per second.
3. High availability for product search.
4. Strong consistency for payment and inventory.
5. Latency approximately 200 ms.
6. Horizontal and vertical scaling support.

Consistency and Availability Considerations

- Product search prioritizes availability.
- Payment and order placement require consistency.
- Inventory must prevent overselling during flash sales.

High Level Design (HLD)

The system consists of Client, Frontend, Backend Services (Product, Cart, Order, Payment, Inventory), and Database layers. High availability is ensured for search, while consistency is enforced for payments and inventory. The system supports horizontal and vertical scaling.



Low Level Design (LLD)

Product Module

- Stores product details and inventory.
- Supports keyword-based search with pagination.

Cart Module

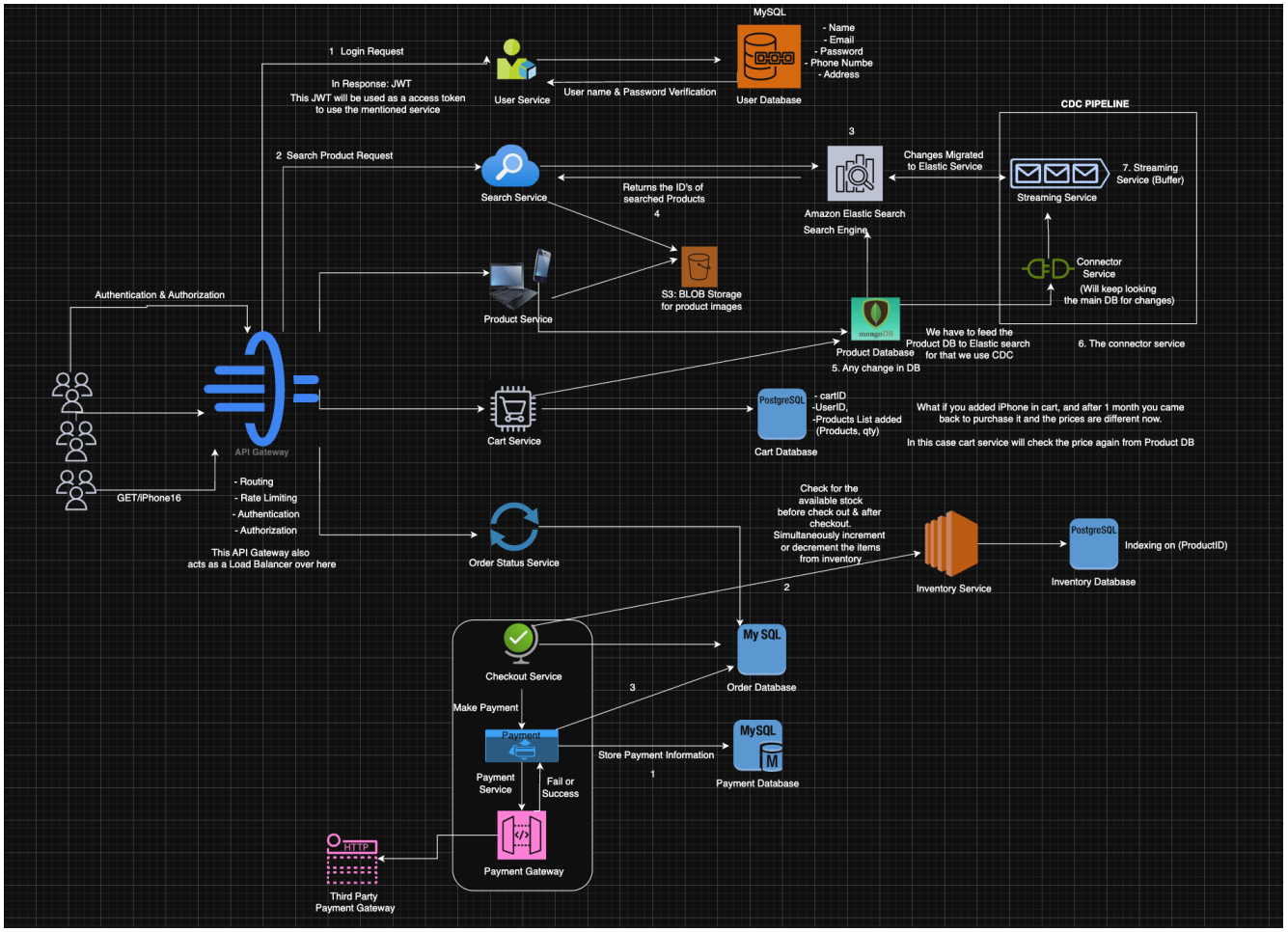
- Maintains user-specific cart.
- Supports add, update, and delete operations.

Order and Payment Module

- Generates order ID after checkout.
- Confirms payment and updates order status.

Inventory and Race Condition Handling

- Manages limited stock.
- Uses locking or transactions during flash sales.



Outcome

- Designed a scalable E-commerce system.
- Identified functional and non-functional requirements.
- Designed REST APIs.
- Understood race conditions and inventory consistency.
- Gained practical system design exposure.